

Praktikumsaufgaben

Aufgabenblatt 1 - Grundbegriffe	2
Aufgabenblatt 2 – Datentypen, Variablen, Konstanten	5
Aufgabenblatt 3 - Operatoren, Ausdrücke, Anweisungen	8
Aufgabenblatt 4 – Kontrollstrukturen	10
Aufgabenblatt 5 – Kontrollstrukturen und einfache Algorithmen	11
Aufgabenblatt 6 – Felder und Strings	13
Aufgabenblatt 7 – Felder und Zeiger	14
Aufgabenblatt 8 – Funktionen	15
Aufgabenblatt 9 – Funktionen, Kommandozeilenparameter	17
Aufgabenblatt 10 - Strukturen und Zeiger	18
Aufgabenblatt 11 – Dynamische Speicherverwaltung	20
Aufgabenblatt 12 – Weitere Operatoren und Datentypen	23
Aufgabenblatt 13 – Anwendung zyklische Warteschlange	25

Aufgabenblatt 1 - Grundbegriffe

Aufgabe 1

Schreiben Sie Ihr erstes C++-Programm, das den Text: „Alles klar!“ auf der Konsole ausgibt. Sie können mit Eclipse CDT über File/new/Project/C++ Project ein neues Projekt erzeugen. Erzeugen Sie zuerst ein „Hello World C++ Project“ und modifizieren Sie den Ausgabertext. Erzeugen Sie dann ein leeres Projekt und schreiben den Code aus dem Gedächtnis (auswendig! Nicht Cut & Paste)

Aufgabe 2

Nehmen sie das unten angegebene Programm.

1. Bringen Sie das Programm zum Laufen. Experimentieren sie, indem sie Änderungen am Programm durchführen.
2. Experimentieren Sie mit Variablen Namen! Finden Sie Variablen Namen, die nicht zulässig sind, d.h. bei denen der Compiler eine Fehlermeldung generiert. Überlegen sie sich, warum bestimmte Variablen Namen nicht zulässig sind. Protokollieren Sie Ihre Versuche.
3. Deklarieren Sie eine neue Variable, lesen Sie einen Wert in die Variable ein (mit cin) und geben Sie den Wert wieder aus (mit cout)
4. Führen Sie das Programm schrittweise im Debugging Modus aus (Zeile für Zeile). Beobachten Sie die aktuellen Werte der Variablen num und f mit dem Debugger. Beobachten Sie die Werte insbesondere vor der Ausführung der Zeile 5, nach der Zeile 9 und vor der Zeile 15

```
1      // k01b06.cpp    F. Kaspar
2      #include <iostream>
3
4      int main(void) {
5          int num;
6          float f;
7
8          std::cout << "Ganzzahl eingeben:";
9          std::cin >> num;
10         std::cout << "Kommazahl eingeben:";
11         std::cin >> f;
12         std::cout << num << ' ';
13         std::cout << f << '\n';
14
15         return 0;
16     }
```

Aufgabe 3

Nehmen sie das unten angegebene Programm.

1. Bringen Sie das Programm zum Laufen. Experimentieren sie, indem sie Änderungen am Programm durchführen.

2. Experimentieren Sie mit den Namen der Funktionen. Finden Sie zulässige und nicht zulässige Namen. Studieren Sie die Fehlermeldungen bei nicht zulässigen Namen. Protokollieren Sie Ihre Versuche.
3. Führen Sie das Programm im Debugging Modus aus. Setzen sie geeignete Breakpoints, z.B. in den Zeilen 10, 11 und 15. Beobachten Sie den Kontrollfluss mit dem Debugger.
4. Fügen Sie neue Funktionen ein, die wie `funk1()` jeweils ein Wort ausgeben. Sie können `funk1()` kopieren und dann ändern. Vergessen Sie die Prototypen nicht. Rufen Sie die Funktionen in `main()` auf. Es soll der Text „Alles Gute! Alles klar! Alles Logo!“ ausgegeben werden.

```
1 // k01b05.cpp F. Kaspar
2 #include <iostream>
3 using namespace std;
4
5 // Prototyp fuer funk1()
6 void funk1(void);
7
8 int main(void) {
9     cout << "Alles ";
10    funk1();
11    cout << "Logo!";
12    return 0;
13 }
14 void funk1(void) {
15     cout << "klar? ";
16 }
```

Aufgabe 4

Betrachten sie das unten angegebene Programm. Versuchen Sie durch Lesen zu verstehen, was das Programm macht. Bringen Sie das Programm zum Laufen. Versehen Sie das Programm an geeigneten Stellen mit Schreibanweisungen zur Ausgabe von Variablen Werten oder Text, um den Ablauf des Programms besser zu verstehen. Hinweis: Geben Sie z.B. nach der Zeile 8 die Variable `i` aus. Geben Sie z.B. nach der Zeile 6 einen Text aus. Ersetzen Sie in Zeile 7 z.B. `" "` durch `"/"`. Experimentieren Sie alternativ mit dem Debugger von Eclipse CDT. Führen Sie das Programm zeilenweise aus. Beobachten Sie die Variable `i` mit dem Debugger: Wie verändert sich der Wert der Variablen `i` nach Ausführung der 5., 7., 8.,... Zeile?

```
1 //a01b01.cpp
2 #include <iostream>
3 using namespace std;
4 int main() {
5     int i=9;
6     do {
7         cout << i << " ";
8         i--;
9     } while (i<0);
10    cout << '\n';
11    return 0;
12 }
```

Aufgabe 5

1. Bauen Sie in das gegebene Programm Fehler ein. Übersetzen Sie das fehlerhafte Programm und studieren Sie die Fehlermeldungen des Compilers. Ersetzen Sie z.B. in Zeile 1 `//` durch `/` oder in Zeile 5 `()` durch `(` (oder lassen Sie in Zeile 7 `std::` weg oder ersetzen Sie in Zeile 9 `"zwei"` durch `'zwei'` oder lassen Sie in Zeile 13 den Strichpunkt weg.
2. Führen Sie den Einbau von Fehlern als Partnerübung durch. Bauen Sie Syntaxfehler ein. Ihr Partner muss die Syntaxfehler finden. i) durch Codeinspektion, d.h. Lesen des Programms. ii) Versuch das Programm zu übersetzen, d.h. den Compiler befragen und die Fehlermeldungen des Compilers zu verstehen. iii) Teile des Programms in Kommentar setzen, um den Text, wo der Fehler verborgen ist, zu verkleinern. iv) Vergleich mit dem vorliegenden Quelltext. Tauschen Sie die Rollen

```
1      // a01b02.cpp ANSI Variante C++
2
3      #include <iostream>
4
5      int main()
6      {
7          std::cout << "eins"
8                  << "\n";
9          std::cout << "zwei"
10                 << '\n';
11          std::cout << "drei"
12                 << std::endl;
13          return 0;
14      }
```

Dieses Aufgabenblatt ist während der ersten Praktikumswoche zu bearbeiten.

Aufgabenblatt 2 – Datentypen, Variablen, Konstanten

Aufgabe 1

Was gibt dieses Programm aus? Überlegen Sie sich die Ausgabe zuerst auf dem Papier.

```
//a02b01.cpp
#include <iostream>
using namespace std;

int main(void) {
    cout << 777 << endl;
    cout << 1.5 * 7 << endl;
    cout << 'A' << endl;
    cout << 'A' + 1 << endl;
    cout << 'A' * 1.5 << endl;
    cout << 'A' * 1 << endl;
    cout << 'A' * 1.0 << endl;
    return 0;
}
```

Hinweis: Konstanten haben einen Typ. Es finden automatische Typumwandlungen beim Auswerten der Ausdrücke statt. cout erkennt den Typ des Ausgabewertes und gibt den Wert im dazugehörigen Format aus.

Aufgabe 2

Was gibt das folgende Programm aus und warum?

```
//a02b02.cpp
#include <iostream>
using namespace std;
int main(void) {
    cout << "Groesse: " << sizeof('0') << " Byte" << endl;
    cout << "Groesse: " << sizeof(1) << " Byte" << endl;
    cout << "Groesse: " << sizeof(1.0) << " Byte" << endl;
    cout << "Groesse: " << sizeof(1.0f) << " Byte" << endl;

    cout << "Groesse: " << sizeof(char) << " Byte" << endl;
    cout << "Groesse: " << sizeof(int) << " Byte" << endl;
    cout << "Groesse: " << sizeof(float) << " Byte" << endl;
    cout << "Groesse: " << sizeof(double) << " Byte" << endl;

    return 0;
}
```

Aufgabe 3

Können Sie die Datei iostream oder die Datei iostream.h auf ihrem Rechner finden? In welchem Verzeichnis befindet sie sich? Ist es eine Textdatei? Können Sie Sie lesen? Verstehen Sie etwas?

Aufgabe 4

Schreiben Sie ein Programm in dem Sie eine Variable für jeden Basisdatentyp aus C++ deklarieren, jeder Variable einen Wert zu weisen und jede Variable ausgeben.

Aufgabe 5

Was gibt dieses Programm aus? Überlegen Sie sich die Ausgabe zuerst auf dem Papier.

```
//a02b03.cpp
#include <iostream>
using namespace std;
void f1(void);
void f2(void);
int zahl = 3; /* globale Variable */

int main(void) {
    int zahl; /* lokal bezüglich main() */
    f1();
    cout << zahl << endl;
    return 0;
}

void f1(void) {
    cout << zahl << endl;
    f2();
}

void f2(void) {
    int zahl = 7; /* lokal bezüglich f2() */
    cout << zahl << endl;
}
```

Denken Sie daran, dass die Sichtbarkeit lokaler Variablen auf die Funktion beschränkt ist in der sie deklariert sind.

Aufgabe 6

Sie haben folgende Variablen Deklarationen:

```
int i; long l; float f; double d;
```

Sie machen folgende Zuweisungen:

- 1) `d=100/3; f=d; l=f; i=l;`
- 2) `d=100/3.0; f=d; l=f; i=l;`
- 3) `d=(float)100/3; f=d; l=f; i=l;`

1. Welche Werte enthalten die Variablen d, f, l, i für jeden der Fälle 1)-3)? Überlegen Sie sich das Ergebnis mit Bleistift und Papier und überprüfen Sie es anschließend mit Hilfe eines Programms. Geben Sie an wo Typumwandlungen stattfinden und von welchem Typ in welchen umgewandelt wird. Unterscheiden Sie Typumwandlung bei der Auswertung des Ausdrucks und bei der Zuweisung.
2. Bei welchen Zuweisungen geht Information verloren (Nachkommastellen, Genauigkeitsverlust)? Gibt es eine Warnung des Compilers?

Aufgabe 7

- 1) Betrachten Sie das unten angegebene Programm. Versuchen Sie durch Lesen zu verstehen, was das Programm macht. Bringen Sie das Programm zum Laufen. Versehen Sie das Programm an geeigneten Stellen mit Schreibanweisungen zur Ausgabe von Variablen Werten oder Kommentaren, um den Ablauf des Programms besser zu verstehen.
- 2) Analysieren Sie das Programm mit dem Debugger. Beobachten Sie den Zustand der Variablen bei der schrittweisen Durchführung des Programms.
- 3) Ändern Sie das Verhalten des Programms durch Änderung der Ganzzahl Konstanten (1, 7, 9, 2, 11), die verwendet werden.
- 4) Verwenden Sie namespace. Lassen Sie dafür std:: weg.
- 5) Ersetzen Sie die Konstanten (1, 7, 9, 2, 11) durch int Variablen, denen Sie die entsprechenden Werte zuweisen. Beispiel: 7 wird ersetzt durch die Ganzzahlvariable anz1, der der Wert 7 zugewiesen wurde.

Quelltext:

```
//k04b07.cpp
#include <iostream>

int main() {
    int i, resultat;
    for (i = 1; i < 7; i++)
        std::cout << i << '\t';
    std::cout << '\n';

    for (i = 9; i > 2; i--) {
        std::cout << i << '\t';
        std::cout << i * i << '\t';
        std::cout << '\n';
    }
    resultat = 0;
    for (i = 1; i < 11; i = i + 2) {
        resultat = resultat + i;
        std::cout << "Zwischensumme: " << resultat << std::endl;
    }
    return 0;
}
```

Aufgabe 8

Schreiben Sie ein Programm in dem Sie die Konstante -3.3 durch explizite Typumwandlung (Cast) in die Typen int, unsigned int, float und bool umwandeln und das Ergebnis auf der Standardausgabe ausgeben.

Dieses Aufgabenblatt ist während der zweiten Praktikumswoche zu bearbeiten.

Aufgabenblatt 3 - Operatoren, Ausdrücke, Anweisungen

Aufgabe 1

Schreiben Sie folgendes Programm: Das Programm soll einen Betrag in Cent als int von der Konsole einlesen und den Betrag in Euro und Cent ausgeben. Z.B. einlesen: 9876 und ausgeben: 98 Euro, 76 Cent. Verwenden Sie Ganzzahldivision / und Divisionsrestoperator %.

Aufgabe 2

Was ergeben diese Ausdrücke, d.h. was ist jeweils der Wert von x?

```
x = 3 - 1 * 7 + 9 % 5;  
x = -3 + 4 * 5 - 6;  
x = 3 + 4 % 5 - 6;  
x = (3 - 1) * ((7 + 9) / 5);  
x = (3 - (1 * 7) + 9) % 5;  
x = 5 - 2 * (7 + 9 % 5);  
x = (3 - 1) * 7 + (9 % 5);  
x = 17 / 5 - 3 * 4;  
x = 17 / (5 - 3) * 4;  
x = 17 / ((5 - 3) * 4);  
x = -3 * ((17 % -6) / 2);
```

Überlegen Sie sich das Ergebnis mit Bleistift und Papier und überprüfen Sie es anschließend mit Hilfe eines Programms.

Aufgabe 3

Ergänzen sie das unten gegebene Programm für die Vergleichsoperationen:

```
i > j, i >= j, i <= j, i == j, i != j  
//k03b01.cpp  
#include <iostream>  
using namespace std;  
  
int main(void) {  
    int i, j;  
    bool b;  
  
    cout << "Bitte eine Zahl eingeben: ";  
    cin >> i;  
    cout << "Bitte eine Zahl eingeben: ";  
    cin >> j;  
    b = i < j;  
    cout << "Bedingung: " << i << " < " << j << " ist: ";  
    if (b)  
        cout << "erfüllt" << endl;  
    else  
        cout << "nicht erfüllt" << endl;  
    /* hier Code für andere Vergleichsoperationen ergänzen */  
  
    return 0;  
}
```


Aufgabe 4

Schreiben Sie ein Programm, in dem Sie den booleschen Ausdruck $(a \text{ XOR } b)$ berechnen. Lesen Sie zwei boolesche Werte in zwei boolesche Variablen a und b ein. Weisen Sie der booleschen Variablen my_xor das Ergebnis von $a \text{ XOR } b$ zu. Geben Sie den Wert von my_xor auf der Konsole aus.

Hinweis: $a \text{ XOR } b = (a \text{ OR } b) \text{ AND NOT } (a \text{ AND } b)$. Verwenden Sie die logischen Operatoren $\&\&$ für AND, $!$ für NOT und $||$ für OR.

Aufgabe 5

Das unten angegebene Programm hat Fehler. Finden Sie die Fehler durch i) lesen des Programms, ii) abtippen und übersetzen und Interpretieren der Fehlermeldungen des Compilers, iii) falls nötig durch Entfernen von Teilen, d.h. reduzieren der Fläche auf der Sie nach Fehlern suchen müssen. Ihr Programm ist nun syntaktisch korrekt, d.h. der Computer akzeptiert es. Arbeiten Sie so, wie Sie es erwarten? Testen Sie das Programm durch Eingabe geeigneter Werte. Testen Sie insbesondere auch den Ausdruck: $i \&\& j$:

```
//k03b03.cpp
#include <iostream>
using namespace std;
int summe(int a, int b);
int produkt(int a, int b);
int main(void) {
    int i, j;
    cout << "Bitte zwei Zahlen eingeben: ";
    cin >> i;

    cout << "i + j = " << summe(i, j) << endl;
    cout << "i * j = " << produkt(i, j) << endl;
    cout << "i ||| j = " << (i ||| j) << endl;
    cout << "i && j = " << (i && j) << endl;
    return 0;
}
int summe(int a, int b) {
    return a + b;
}
int produkt(int a, int b) {
    return;
}
```

Aufgabe 6

Schreiben Sie ein Programm, das eine Ganzzahl von der Konsole einliest und prüft ob die Zahl gerade ist.

Aufgabe 7

Schreiben Sie ein Programm, das zwei Ganzzahlen von der Konsole einliest und prüft, ob die erste Zahl ohne Rest durch die zweite Zahl teilbar ist. Hinweis: $\%$ Operator verwenden.

Dieses Aufgabenblatt ist während der dritten Praktikumswoche zu bearbeiten.

Aufgabenblatt 4 – Kontrollstrukturen

Aufgabe 1

Summieren Sie mit der while Schleife die Zahlen 1 bis 100 auf und geben Sie das Ergebnis an der Konsole aus.

Aufgabe 2

Berechnen Sie mit der do while Schleife den Wert von 8! (Fakultät, $0! = 1$, $1! = 1$, $2! = 1 \cdot 2$, ..., $8! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8$) Geben Sie das Ergebnis auf der Konsole aus.

Aufgabe 3

Geben Sie mit der for Schleife die ersten 20 Zahlen der Fibonacci Reihe aus ($f_{n+2} = f_{n+1} + f_n$; $f_0 = 0$, $f_1 = 1$).

Aufgabe 4

In C gibt es die Funktion rand(), die eine Zufallszahl vom Typ int liefert. Geben Sie mit Hilfe der Funktion rand() und einer Wiederholungsschleife Ihrer Wahl 10 Integer-Zufallszahlen auf der Konsole aus. Damit die Verwendung der Funktion rand() möglich ist, müssen sie oben in ihr Programm

```
#include <stdlib.h>
```

eingeben. Eventuell müssen Sie die Option -lm beim Übersetzen (genauer Linken) des g++ angeben.

Aufgabe 5

Im ASCII Code ist die Differenz zwischen Kleinbuchstaben und Großbuchstaben 32 (Bsp.: 'A' ist dezimal 65, 'a' ist 97 dezimal). Um einen Kleinbuchstaben in einen Großbuchstaben umzuwandeln, kann man also einfach 32 abziehen. Die Differenz kann man auch als ('a' - 'A') ausdrücken. Das dokumentiert besser den Sinn des Quelltextes als die „magische Zahl“ 32. Schreiben Sie ein Programm, das ein Zeichen von der Tastatur einliest und Kleinbuchstaben als Großbuchstaben ausgibt. Andere Zeichen als Kleinbuchstaben werden nicht verändert.

Aufgabe 6

Schreiben Sie ein Programm, das die Primzahlen zwischen min und max ermittelt und auf der Konsole ausgibt. Lesen Sie min und max von der Konsole ein. Hinweis: Verwenden Sie die Ganzzahldivision bzw. den Divisionsrestoperator. Dokumentieren Sie, wie Sie das Programm getestet haben.

Der Abgabetermin für dieses Aufgabenblatt wird separat bekannt gegeben.

Aufgabenblatt 5 – Kontrollstrukturen und einfache Algorithmen

Aufgabe 1

Schreiben Sie ein Programm, das folgendes Menü für die Steuerung einer Administrationaufgabe realisiert. Punkte des Menüs sind:

i - initialisieren
z - zurücksetzen
s - starten
d – drucken
e - beenden

Verwenden Sie die switch Anweisung. Vorgehen:

1. Geben Sie durch eine Funktion die obigen Punkte als Menütex aus.
2. Lesen Sie den Buchstaben für die Menüsteuerung ein und werten Sie ihn in der switch Anweisung aus.
3. Fügen Sie im Punkt default der switch Anweisung die Behandlung ungültiger Eingaben ein, d.h. Angabe eines falschen Buchstabens.
4. Erweitern Sie die Auswertung eingegebener Buchstaben, so dass außer dem Kleinbuchstaben auch der entsprechende Großbuchstabe gültig ist (Eingabe von s und S führen zur Aktion starten).
5. In einer ersten Variante beenden Sie das Programm nach einer Auswertung eines Buchstaben in der switch Anweisung. In einer zweiten Variante fügen Sie eine Wiederholungsschleife um die switch Anweisung, sodass Sie das Menü beliebig oft durchlaufen können.

Es genügt, wenn Sie die angewählte Aktion im jeweiligen switch Punkt als Text ausgeben. z.B.:

```
cout << "starten" << endl;
```

Aufgabe 2

Mit dem angegebenen Verfahren können Sie die Anzahl Bit für die Darstellung eines

```
unsigned short int  
unsigned int  
unsigned long int
```

ermitteln. Gehen Sie folgendermaßen vor: Initialisieren Sie eine Variable von einem der oben gegebenen Typen mit 1. Multiplizieren Sie dann solange mit 2 bis die Zahl 0 wird. Zählen Sie, wie oft Sie mit 2 multipliziert haben (Erklärung: Multiplikation mit 2 entspricht Bitshift nach links). Vergleichen Sie ihr Ergebnis mit dem Ergebnis, wenn Sie den sizeof Operator verwenden. Funktioniert das Verfahren auch, wenn Sie statt `unsigned int` als Datentyp `int` verwenden?

Aufgabe 3

Schreiben Sie ein Programm, das die Schaltjahre von 1900 bis heute ausgibt. Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl durch 4, aber nicht durch 100 teilbar ist. Sollte die Jahreszahl durch 400 teilbar sein, handelt es sich dennoch um ein Schaltjahr.

Aufgabe 4

Wir betrachten die Funktion $\text{ld}(x)=\log_2(x)$. Für diese Funktion gibt es keine Bibliotheksfunktion in ANSI C oder ANSI C++. Sie können sie jedoch selbst erzeugen durch Verwendung der Beziehung;

$$\log_2(x)=\log_{10}(x)/\log_{10}(2)$$

und der in ANSI C und ANSI C++ vorhandenen Bibliotheksfunktion `log10()`:

```
double log10(double x);
```

Das Argument für die Funktion `log10` ist vom Typ `double` und der Rückgabewert ist auch vom Typ `double`.

Vergessen Sie nicht den Eintrag (ANSI C++ mit Namensräumen)

```
#include <cmath>
```

am Anfang Ihres Programms, wenn Sie `log10()` verwenden!

Schreiben Sie ein Programm, das eine Ganzzahl in eine Variable vom Type `unsigned long int` einliest. Das Programm soll die Anzahl Bit ermitteln, die mindestens zur Darstellung der Zahl notwendig ist. Die Formel zur Ermittlung der Anzahl Bit für die Zahl n ist: $\lceil \log_2(n) \rceil$. $\lceil x \rceil$ bezeichnet den nächst größeren Ganzzahlwert größer oder gleich x . Geben Sie das Ergebnis auf der Konsole aus. Erklären Sie, wie Sie Ihr Programm getestet haben.

Aufgabe 5

Schreiben Sie ein Programm, das folgende Aufgabe löst:

Auf einer Wiese hütet die Liesel ihre Gänse und anderes Kleinvieh. Der Pfarrer der gerade vorbeikommt fragt Liesel nach der Anzahl ihrer Tiere. Liesel antwortet: "Ich hüte doppelt so viele Gänse wie Hühner, aber dreimal so viele Kaninchen wie Schafe. Insgesamt haben meine Tiere 90 Beine." Wie viele Tiere in welcher Gattung hütet Liesel? (Hinweis: jedes Tier hat mindestens 2 Beine, also gibt es insgesamt nicht mehr als 45 Tiere, also nicht mehr als 45 Tiere von jeder Art. Also durchprobieren: 0, 1, ..., 45 Gänse, ... und Nebenbedingung prüfen. Lösung: 14 Gänse; 9 Kaninchen; 7 Hühner und 3 Schafe.)

Aufgabe 6

Schreiben Sie ein Programm, das die Rückgabe von Wechselgeld nach einem Kauf bestimmt. Das Programm soll zwei Zahlen einlesen: den Kaufbetrag und den vom Kunden dem Kassierer gegebenen Betrag. Das Programm soll das Wechselgeld berechnen und dabei soll eine möglichst geringe Anzahl von Banknoten und Münzen zurückgegeben werden. Gehen Sie von der Euro-Stückelung aus (1 Cent, 2 Cent, 5 Cent, 10 Cent,...500 €). Prüfen Sie, ob der Kaufbetrag nicht größer ist als der gegebene Betrag. Geben Sie die ermittelte Stückelung des Wechselgelds auf der Standardausgabe aus.

Der Abgabetermin für dieses Aufgabenblatt wird separat bekannt gegeben.

Aufgabenblatt 6 – Felder und Strings

Aufgabe 1

Programmieren Sie einen Stapel (Stack) für Ganzzahlwerte (int). Stapel bedeutet, das zuletzt eingefügte Element wird zuerst entfernt. Verwenden Sie ein int Feld konstanter Länge. Beginnen Sie mit dem Einfügen an der Position 0 des int Feldes. Eine Variable zur Verwaltung der aktuellen Anzahl Elemente im Stack genügt. Dadurch ist die Einfüge Position und das als erstes heraus zu nehmende Element bestimmt. Berücksichtigen Sie die Fälle, dass bei einem leeren Stapel kein Element entfernt werden kann und bei einem vollen Stapel kein Element eingefügt werden kann. Verwenden Sie ein Menü zur Steuerung Ihres Stapels. Wie haben Sie Ihr Programm getestet?

Aufgabe 2

Schreiben Sie ein Programm zur Ermittlung der ersten 1000 Primzahlen. Merken Sie sich die ermittelten Primzahlen in einem Feld der Dimension 1000. Beginnen Sie mit der Primzahl 2 (Wert von Feldelement 0). Prüfen Sie alle Zahlen beginnend mit 3. Prüfen Sie jede neue Zahl ob sie prim ist, indem Sie durch jede bisher ermittelte Primzahl dividieren. Ist eine Zahl durch keine bisher ermittelte Primzahl teilbar ist sie eine Primzahl. Fügen Sie jede ermittelte Primzahl in das Feld ein. Das Programm soll die größte ermittelte Primzahl ausgeben.

Aufgabe 3

Lesen Sie eine Zeichenkette in eine Variable vom Typ string der C++ Klassenbibliothek ein.

1. Bestimmen Sie die Anzahl Vokale (a, e, i, o, u) in der Zeichenkette. Es genügt der Fall der Kleinbuchstaben ohne Umlaute. Benutzen Sie die string Variable wie ein Feld und greifen Sie auf die einzelnen Zeichen in Feldschreibweise zu.
2. „Verschlüsseln“ Sie den Text der string Variable indem Sie die Buchstabenwerte folgendermaßen ersetzen: $a \rightarrow b$, $b \rightarrow c$, ..., $y \rightarrow z$, $z \rightarrow a$. Beispiel: aus hal wird ibm. Geben Sie das Resultat auf der Konsole aus.
3. Schreiben Sie den „Entschlüsselungscode“ für 2. Das heißt nach „Verschlüsselung“ und „Entschlüsselung“ erhalten Sie den ursprünglichen Text.
4. Denken Sie sich ein eigenes Verschlüsselungsverfahren aus. Geben Sie die Verschlüsselung und die Entschlüsselung (Umkehroperation) an.

Aufgabe 4

Lösen Sie die Aufgabenteile 1 bis 4 der Aufgabe 3 durch Verwendung eines Null terminierten char Feldes. D. h. schreiben Sie Ihren Code aus Aufgabe 3 so um, dass Sie statt einer Variable vom Typ string ein Null terminiertes char Feld verwenden.

Bearbeiten Sie alle Aufgaben. Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 7 – Felder und Zeiger

Aufgabe 1

Schreiben Sie ein Programm, bei dem Sie ein Ganzzahlfeld der Dimension `dim` deklarieren, die Feldelemente initialisieren und dann die Werte um eine Position nach rechts rotieren. Das heißt der Wert des Feldelements 0 wird an Position 1 gespeichert, der Wert des Feldelements 1 wird an Position 2 gespeichert,..., der Wert des Feldelements `dim-1` wird an Position 0 gespeichert. Verwenden Sie die Zeigerschreibweise (z.B.: `*(ifeld+1)` und nicht `ifeld[1]`). Falls Sie mit der Zeigerschreibweise nicht vertraut sind entwickeln Sie den Code am besten zuerst in Feldschreibweise und ändern Sie in einem zweiten Schritt den Code in Zeigerschreibweise um.

Aufgabe 2

Schreiben Sie Programme für die unten beschriebenen Aufgaben. Es handelt sich um die Verarbeitung von Zeichenketten, die als C-Strings repräsentiert sind (Null terminierte char Felder). Verwenden Sie die Pointer Schreibweise (z.B.: `*(s+1)` und nicht `s[1]`). Falls Sie mit der Zeigerschreibweise nicht vertraut sind entwickeln Sie den Code am besten zuerst in Feldschreibweise und ändern Sie in einem zweiten Schritt den Code in Zeigerschreibweise um. Verwenden Sie keine Bibliotheksfunktionen. Es sollen folgende Funktionalitäten in Programmen implementiert und durch geeignete Ein- und Ausgaben demonstriert werden:

1. Schreiben sie Ihren Code aus Aufgabe 4 des Aufgabenblattes 6 in Zeigerschreibweise um.
2. Lesen Sie von der Konsole einen Text in einen char Feld ein. Wandeln Sie alle in dem Feld, d.h. dem C-String enthaltenen Kleinbuchstaben in Großbuchstaben um. Also: Falls es sich bei einem Zeichen um einen Kleinbuchstaben handelt soll das Programm ihn in einen Großbuchstaben umwandeln. Handelt es sich um ein anderes Zeichen soll das Zeichen nicht geändert werden. Hinweis: Für die Umwandlung von einem Kleinbuchstaben in einen Großbuchstaben müssen Sie von dem Zeichen die Differenz zwischen Kleinbuchstaben und Großbuchstaben abziehen (`'a'-'A'`)
3. Lesen Sie einen Text, der eine natürliche Zahl darstellt in ein char Feld ein. Wandeln Sie diesen Text, d.h. den C-Strings der eine natürliche Zahl darstellt in seinen Integerwert um. Negative Zahlen werden nicht berücksichtigt. Falls die Zeichenkette ein Zeichen ungleich den Ziffern 0,1,...,9 enthält, soll eine Fehlermeldung ausgegeben werden und das Programm beendet werden. Beispiel: Ich tippe 37 auf der Tastatur, d.h. der Text "37" wird eingelesen. Mit dem Ausdruck: `i = 10*('3'-'0')+'7'-'0'`; erhalte ich den Integerwert 37 aus dem Text "37". Falls der Fall mit einer beliebigen Anzahl Zeichen für Sie zu schwierig ist, lösen Sie den Fall mit vier Zeichen, d.h. einer Zahl mit 4 Ziffern.

Rotieren Sie die Elemente eines C-String um 1 Stelle nach rechts. D.h. das Zeichen an der Stelle `i` kommt an die Stelle `i+1`. Das Zeichen an der Stelle `i=strlen(s)-1` wird an die nullte Stelle geschoben. Das terminierende 0-Zeichen wird nicht rotiert. Verwenden Sie eine Hilfsvariable (`char temp;`) analog dem Tausch der Inhalte zweier Variablen. Kein Hilfsfeld verwenden!

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 8 – Funktionen

Aufgabe 1

1. Schreiben Sie ein Programm mit einer Funktion `produkt()`. Die Funktion `produkt` soll 2 `int` Argumente haben und einen Rückgabewert vom Typ `int`. Sie soll das Produkt der übergebenen Parameter berechnen und als Rückgabewert zurückgeben. Demonstrieren Sie die Funktionalität, indem Sie die Parameter von der Tastatur einlesen und das Ergebnis auf der Konsole ausgeben.
2. Wandeln Sie das obige Programm ab, indem Sie den Typ `int` durch den Typ `float` ersetzen. Das heißt Sie lesen `float` Werte ein, übergeben `float` Werte an die Funktion `produkt()` und die Funktion `produkt()` gibt einen Wert vom Typ `float` zurück.
3. Können Sie die beiden Funktionen aus 1. und 2. mit dem gleichen Funktionsnamen in einem Programm verwenden? Testen Sie folgende Fälle:

Aufruf: `produkt(7,5)` und `produkt(1.5f,8.0f)`

Aufruf: `produkt(7,5)` und `produkt(1.5,8.0)`

Aufruf: `produkt(7,5.0f)` und `produkt(7f,5)`

Aufgabe 2

Schreiben Sie eine Funktion, die die Zahlen $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, ... aufsummiert. Entwickeln Sie die Funktion in zwei Varianten:

1. Die Funktion erhält als Übergabeparameter eine Ganzzahl, die angibt, wie viele Summanden für die Summe berücksichtigt werden sollen. Zum Beispiel bedeutet die Übergabe von 20, dass der letzte berücksichtigte Summand $\frac{1}{2^{20}}$ ist.
2. Die Funktion erhält als Übergabeparameter einen Wert für die Genauigkeit (z.B. `epsilon=0.001`) mit der die Summe berechnet werden soll. Es soll so lange summiert werden, bis gilt `summand < epsilon`, wobei `summand = $\frac{1}{2^n}$` , mit `n=1,2,3,4,5,...`

In beiden Fällen soll die Funktion den berechneten Summenwert als Rückgabewert zurückgeben. Führen sie die beiden Funktionen mit einem geeigneten Hauptprogramm vor.

Aufgabe 3

In früheren Aufgaben haben Sie Programme für verschieden Aufgaben erstellt. zur Zeichenkettenverarbeitung erstellt. Wandeln Sie diese Programme in Funktionen um.

Wandeln Sie das Programm von Aufgabe 3, Teil 1 aus Aufgabenblatt 6 in eine Prozedur um. Der Prototyp der Prozedur:

```
int anz_vokale(char *s);
```

Wandeln Sie das Programm von Aufgabe 2, Teil 2 aus Aufgabenblatt 7 in eine Prozedur um. Der Prototyp der Prozedur:

```
void to_upper (char *s);
```

Wandeln Sie das Programm von Aufgabe 2, Teil 3 aus Aufgabenblatt 7 in eine Prozedur um. Der Prototyp der Prozedur:

```
long int my_toint (char *s);
```

Wandeln Sie das Programm von Aufgabe 2, Teil 0 aus Aufgabenblatt 7 in eine Prozedur um. Der Prototyp der Prozedur:

```
void my_rotate(char *s);
```

Aufgabe 4

Programmieren und testen Sie die Funktion:

```
void my_encode (char *out, const char *in)
```

my_encode verschlüsselt den als 2. Argument übergebenen String und speichert ihn im als 1. Argument übergebenen String. Verschlüsseln Sie, indem Sie von jedem Zeichen den Wert 1 abziehen (gleiche Methode wie Aufgabenblatt 6, Aufgabe 3, Teil 2)

Aufgabe 5

Gegeben ist ein klassisches Tauschbeispiel mit Pointern. Formulieren Sie das Beispiel um für Referenzparameter:

```
//a08b01.cpp
#include <iostream>
using namespace std;
void tausch(int *i, int *j);
int main(void) {
    int i1, i2;
    i1 = 111;
    i2 = 777;
    cout << "i1: " << i1
         << " i2: " << i2 << endl;
    tausch(&i1, &i2);
    cout << "i1: " << i1
         << " i2: " << i2 << endl;
    return 0;
}
void tausch(int *i, int *j) {
    int temp;
    temp = *i;
    *i = *j;
    *j = temp;
}
```

Aufgabe 6

Programmieren Sie eine Funktion zur Ziehung der Lotto Zahlen. Deklarieren Sie dazu ein Feld als globale Variable der Dimension 49. Initialisieren Sie das Feld mit der Funktion reset() mit den Zahlen von 1 bis 49. Die Feldelemente repräsentieren die Kugeln. Das Element des Feldes mit dem Index 0 enthält die Kugel mit der Ziffer 1, bei Index 1 ist die Ziffer 2, usw. Schreiben Sie eine Funktion ziehe(), die eine Kugel zieht, d.h. eine Zahl von 1 bis 49 liefert. Erzeugen Sie dazu in der Funktion ziehe() eine Zufallszahl von 0 bis 48. Verwenden Sie dafür die rand() Funktion. Das ist der Index der Kugel. Beim ersten Durchgang liefert die Ziehung eine Kugel von 1 bis 49. Entfernen Sie die nach der Ziehung die Kugel aus dem Feld, d.h. verschieben Sie alle Elemente mit einem Index größer als die gezogene Kugel um eins nach links. Die Funktion vermerkt die Anzahl verbliebener Kugeln (nach der ersten Ziehung 48) in der globalen Variablen anzahlKugeln und gibt den Wert der ersten Kugel als Rückgabewert zurück. Beim nächsten Aufruf liefert die Funktion die zweite Kugel. Usw. Bei der Ziehung der Lottozahlen wird nach der Ziehung von 6 Zahlen aufgehört. So machen Sie das auch. D. h. Sie rufen die Funktion ziehe() 6 mal auf.

Aufgabe 7

Ein klassisches Beispiel für eine rekursive Funktion ist die Fakultät:

fakultaet(n+1) = (n+1) * fakultaet(n), fakultät(0)=1. Es gilt n>=0. Programmieren Sie die Fakultät über eine rekursive Funktion.

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. und durchgeführte Tests. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 9 – Funktionen, Kommandozeilenparameter

Aufgabe 1

Schreiben Sie ein Programm, das Argumente von der Kommandozeile entgegennimmt und die Anzahl der Argumente auf der Konsole ausgibt.

Aufgabe 2

In einer früheren Aufgabe haben Sie Fibonacci-Zahlen iterativ berechnet. Berechnen Sie jetzt die n-te Fibonacci-Zahl rekursiv. Verwenden Sie dazu die rekursive Definition der Fibonacci-Zahlen: $f(n) := f(n-1) + f(n-2)$ mit den Abbruchbedingungen $f(1)=1$ und $f(0)=0$;

Aufgabe 3

Ein Schachbrett wird dargestellt als ein zweidimensionales Feld der Dimension 8x8 von booleschen Elementen. Auf dem Schachbrett steht ein Springer auf der Feldposition (startX,startY). $0 \leq \text{startX} \leq 7$ und $0 \leq \text{startY} \leq 7$. Es soll ein Weg des Springers zum Ziel mit der Position (zielX,zielY) gefunden werden. Der Weg besteht aus einer Folge von gültigen Springerzügen. Gültige Springerzüge sind:

3. 1) $x \rightarrow x+2, y \rightarrow y+1$ 2) $x \rightarrow x+2, y \rightarrow y-1$ 3) $x \rightarrow x-2, y \rightarrow y+1$ 4) $x \rightarrow x-2, y \rightarrow y-1$
4. 5) $x \rightarrow x+1, y \rightarrow y+2$ 6) $x \rightarrow x+1, y \rightarrow y-2$ 7) $x \rightarrow x-1, y \rightarrow y+2$ 8) $x \rightarrow x-1, y \rightarrow y-2$

Schreiben Sie dazu eine Funktion, die Springerzüge auf einem Schachbrett, repräsentiert durch das zweidimensionale Feld, durchführt. Die Funktion soll Zugriff auf das globale, zweidimensionale Feld haben. Die Funktion hat zwei Ganzzahl Eingabeparameter, die die Position des Springers auf dem Spielfeld angeben. Die Funktion soll zuerst die Gültigkeit der angegebenen Position prüfen ($0 \leq \text{posX} \leq 7, 0 \leq \text{posY} \leq 7$). Falls die Position außerhalb des Feldes liegt, soll die Funktion mit `return false` beendet werden. Falls die Position (zielX,zielY) erreicht wird soll die Funktion mit `return true` beendet werden. Wird ein Feldelement erreicht mit dem Wert `true` (Schleife) soll die Funktion mit `false` beendet werden. Sonst wird der Wert an der aktuellen Position im Feld auf `true` gesetzt und die Funktion `meinSpringerzug()` ruft die `meinSpringerzug()` Funktion mit allen möglichen Zügen auf (siehe oben).

Einen Zug durchführen heißt, im rekursiven Aufruf die neue Position anzugeben, z.B.:

```
meinSpringerzug(posX+2, posY+1);
```

Schreiben Sie eine `main()` Funktion, die ihre Springerzugfunktion mit einem von der Konsole eingelesenen Startposition (z.B.: 1 6) und einer Zielposition (z.B.: 5 3) aufruft.

Beobachten Sie den Ablauf des Programms mit dem Debugger. Beantworten Sie die Fragen:

1. Bricht bei Ihrem Programm die Rekursion auf jeden Fall ab?
2. Findet Ihr Programm immer einen Weg?
3. Wird ein Weg oder werden mehrere Wege gefunden?
4. Falls mehrere Wege gefunden werden: welcher Weg wird zuerst gefunden?

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 10 - Strukturen und Zeiger

Aufgabe 1

Schreiben Sie ein Programm, das den grundsätzlichen Aufbau einer Struktur demonstriert. Die Struktur soll eine Charakter Variable, eine Integer Variable, eine Float Variable und eine Variable vom Typ string beinhalten. Initialisieren Sie die Elementvariablen der Struktur mit vernünftigen Werten. und geben Sie die Werte anschließend zum Test auf der Konsole aus. Schreiben Sie eine Funktion, die die Ausgabe der Strukturvariablen auf die Konsole übernimmt.

Aufgabe 2

Gegeben ist die Datenstruktur:

```
struct grid {  
    int ix;  
    int iy;  
};
```

Programmieren sie die Funktion vertauschen:

```
void vertauschen(grid * koord1, grid *koord2);
```

die die Werte der Parameter vertauscht. Testen Sie mit einem geeigneten Hauptprogramm.

Aufgabe 3

Es soll eine Software zur Unterstützung einer Wohnungsverwaltung entwickelt werden. Der Datentyp Whg dient als Basis zur Verwaltung von Wohnungen.

```
struct Whg {  
    int zimmer; //Anzahl Zimmer der Wohnung  
    bool frei; //auf true, wenn die Wohnung frei ist  
    float qm; //Groesse der Wohnung in Quadratmeter  
};
```

1. Schreiben Sie die Funktion istFrei(), die den Status der Whg x (frei) als Rückgabewert zurückgibt. Der Prototyp der Funktion ist:

```
bool istFrei(Whg x);
```

2. Schreiben Sie die Funktion setzeBelegt(), die den Status der Whg auf belegt (nicht frei) setzt. Der Prototyp der Funktion ist:

```
void setzeBelegt(Whg & x);
```

3. Schreiben Sie die Funktion setzeFrei(), die den Status der Whg auf frei setzt. Der Prototyp der Funktion ist:

```
void setzeFrei(Whg * xp);
```

4. Schreiben Sie die Funktion sucheFreieWhg(). Die Funktion erhält als Argument ein Feld von Whg Strukturen. Jede Struktur repräsentiert eine Whg. dim gibt die Größe des Feldes an, d.h. die Anzahl Wohnungen. Die Funktion soll die größte freie Wohnung (Struktur vom Typ Whg) mit der angegebenen Anzahl Zimmer zurückgeben. Der Ausnahmefall, dass keine geeignete Whg zur Verfügung steht, wird dadurch behandelt,

dass eine Whg Variable mit 0 Zimmern zurückgegeben wird. Der Prototyp der Funktion ist:

```
Whg sucheFreieWhg(Whg f[], int dim, int anzZimmer);
```

5. Schreiben Sie die Funktion `neueWhg()`. Die Funktion liefert eine initialisierte Struktur vom Typ `Whg` zurück. Die Wohnung ist frei, Anzahl Zimmer und Wohnfläche werden als Argumente übergeben. Der Prototyp der Funktion ist:

```
Whg neueWhg(int zimmer, float qm);
```

6. Schreiben Sie die Funktion `istGroesser()`. Die Funktion soll den Wahrheitswert „wahr“ zurückgeben, wenn die Wohnung `x1` eine größere Fläche hat als Wohnung `x2`, sonst „falsch“. Der Prototyp der Funktion ist:

```
bool istGroesser(Whg * x1, Whg * x2);
```

7. Schreiben Sie die alternative Form der Funktion. Die Funktionalität so die gleiche sein wie in 6. Der Prototyp der Funktion ist:

```
bool istGroesser(Whg & x1, Whg & x2);
```

8. Schreiben Sie eine `main()` Funktion, mit der Sie alle Funktionen demonstrieren.

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 11 – Dynamische Speicherverwaltung

Aufgabe 1

Schreiben Sie das im alten C-Stil mit malloc und free formulierte Programm um in den moderneren C++ Stil mit new und delete. Ersetzen Sie die Ausgabe mit printf durch die Ausgabe mit cout. Brauchen Sie alle vorhandenen include Anweisungen? Brauchen Sie neue include Anweisungen?

```
//a11b01.cpp
#include <stdlib.h>
#include <stdio.h>
int main(void) {
    int * ip;
    ip = (int *) malloc(100000 * sizeof(int));
    if (!ip) {
        printf("Fehler: kein Speicherplatz! \n");
        exit(1);
    }
    free(ip);
}
```

1. Testen Sie den Ausnahmefall, indem Sie die Größe des allokierten Speicherplatzes so weit erhöhen, bis kein weiterer Speicherplatz mehr verfügbar ist.
2. Was geschieht bei Verwendung der Standardversion von new im Ausnahmefall, d.h. wenn der Speicherplatz nicht zur Verfügung gestellt werden kann? Analysieren Sie mit dem Debugger den Kontrollfluss im Ausnahmefall.
3. Können Sie mit new die Ausnahmebehandlung wie mit malloc durchführen, d.h. abfragen ob der Wert von ip ungleich 0 ist? Hinweis: Stichwort nothrow. Überprüfen Sie den Kontrollfluss durch geeignete Schreibanweisungen.

Aufgabe 2

Schreiben Sie das im alten C-Stil formulierte Programm um in den moderneren C++ Stil mit new und delete statt malloc und free. Ersetzen Sie die Ausgabe mit printf durch die Ausgabe mit cout.

```
//a11b02.cpp
#include <stdlib.h>
#include <stdio.h>
int main(void) {
    int i, *ip;
    ip = (int *) malloc(10 * sizeof(int)); /* kein Speicher? */
    for (i = 0; i < 10; i++)
        ip[i] = i + 1;
    for (i = 0; i < 10; i++)
        printf("%d ", ip[i]);
    printf("\n");
    free(ip);
}
```

Nehmen Sie das Programm als Ausgangspunkt, um ein neues Programm im C++ Stil, d.h. mit new und delete zu schreiben, in dem ein Feld der Länge 7 mit Elementen vom Typ

string allokiert wird. string ist der Klassentyp string aus der Standardbibliothek. Lesen Sie Zeichenketten von der Konsole ein und initialisieren Sie damit Ihre Stringvariablen.

Aufgabe 3

Schreiben Sie das im alten C-Stil formulierte Programm um in den moderneren C++ Stil mit new und delete statt malloc und free. Ersetzen Sie die Ausgabe mit printf durch die Ausgabe mit cout. Brauchen Sie noch alle vorhandenen include Anweisungen? Brauchen Sie neue include Anweisungen? Was macht das Programm? Studieren Sie den Kontrollfluss mit Hilfe des Debuggers. Setzen Sie entsprechende Breakpoints.

```
//a11b03.cpp
#include <stdlib.h>
#include <stdio.h>

struct knoten {
    int nummer;
    knoten * links;
    knoten * rechts;
};

void drucken(knoten * start);
knoten * wachsen(int);

int main(void) {
    knoten * kp;
    kp = wachsen(2);
    drucken(kp);
    return 0;
}

knoten * wachsen(int tiefe) {
    static int nummer = 0;
    knoten * kp;
    kp = (knoten *) malloc(sizeof(knoten));
    kp->nummer = ++nummer;
    if (tiefe > 0) {
        kp->links = wachsen(tiefe - 1);
        kp->rechts = wachsen(tiefe - 1);
    } else {
        kp->links = 0;
        kp->rechts = 0;
    }
    return kp;
}

void drucken(knoten * start) {
    printf("Nr.: %d ", start->nummer);
    if (start->links) {
        drucken(start->links);
    }
    if (start->rechts) {
        drucken(start->rechts);
    }
}
```

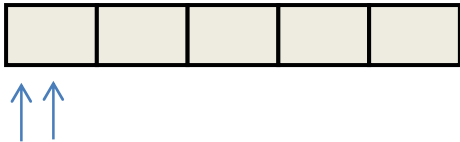
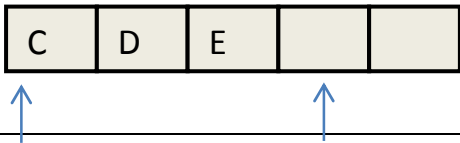
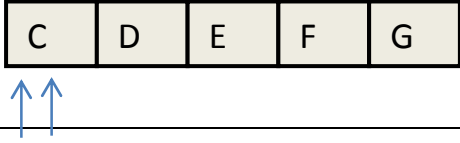
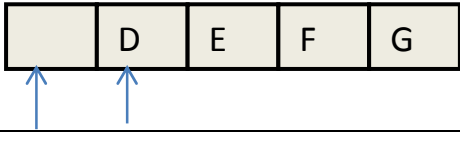
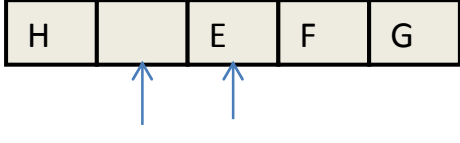
Die Funktion `wachsen()` hat eine lokale Variable vom Typ `knoten *`. Versuchen Sie diese Variable mit dem Debugger zu überwachen. Wie ändert sich der Wert? Wie oft gibt es diese Variable? Wo im Speicher ist diese Variable/sind diese Variablen?

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. und durchgeführte Tests. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 12 – Weitere Operatoren und Datentypen

Aufgabe 1

Programmieren Sie eine zyklische Warteschlange für Charakter. Verwenden Sie ein char Feld konstanter Länge (z.B. 6). Gehen Sie folgendermaßen vor: Benutzen Sie zwei Feldindizes (`int posinsert, posremove;`) um die Einfüge Position und die Entfernen Position zu merken und eine int Variable `anz`, um die Anzahl Elemente in der Warteschlange zu verfolgen. Ist die Warteschlange leer (`anz=0`) kann nicht entfernt werden, ist sie voll kann nicht eingefügt werden. Eingefügt wird an der Position `posinsert`, dann wird der Feldindex `posinsert` erhöht. Entfernt wird an der Position `posremove`.

Zu Beginn ist die Warteschlange leer und <code>posinsert=posremove=anz=0</code> .	
Nach drei Einfüge Vorgängen: <code>posinsert=3 posremove=0</code> <code>anz=3</code>	
Nach fünf Einfüge Vorgängen: <code>posinsert=0 posremove=0</code> <code>anz=5</code>	
Nach einem Lösch Vorgang: <code>posinsert=0 posremove=1</code> <code>anz=4</code>	
Nach einem Lösch und einem Einfüge Vorgang: <code>posinsert=1 posremove=2</code> <code>anz=4</code>	

Testen Sie Ihre Warteschlange möglichst systematisch. Welche Fälle haben Sie berücksichtigt?

Aufgabe 2

Schreiben Sie ein Programm, das eine char Variable bitweise ausgibt. Verwenden Sie dazu die Bit-Operatoren. Das Leerzeichen (ASCII Code 32):

`char ch=' '` wird z.B. als 00100000 ausgegeben.

Hinweise zum Vorgehen: Deklarieren Sie eine char Variable zur Herausprojektion einzelner Bits. Initialisieren Sie die Variable mit 1. Durch den & Operator können Sie damit das niederwertigste Bit aus der char Variablen heraus projizieren. Prüfen Sie das

Ergebnis. Ist das Ergebnis ungleich Null befindet sich an der niederwertigsten Stelle der char Variablen ein 1-Bit. Geben Sie dann eine 1 aus, sonst eine 0. Verschieben Sie das Bit der Variable zur Herausprojektion einzelner Bits mit dem Bitshift Operator um eine Stelle nach links. Projizieren Sie damit das nächste Bit aus der char Variablen heraus. Prüfen Sie auf diese Weise in einer Wiederholungsschleife alle Bit der char Variablen.

Aufgabe 3

Studieren Sie das unten gegebene Programm. Was macht die Funktion transform() und warum? Bringen Sie das Programm zum Laufen und untersuchen Sie das Verhalten mit dem Debugger. Testen Sie die Funktion transform(). Dabei soll mindestens jeder Code einmal ausgeführt werden. Steht Ihnen keine Umgebung zur Verfügung, bei der ein int die Größe von zwei Byte hat, variieren Sie den Ganzzahltyp von int nach short int. Recherchieren Sie zum besseren Verständnis zum Stichwort „little endian“ bzw. „big endian“

```
//a13b01.cpp
#include <iostream>

typedef unsigned char byte;
int transform(int i);

int main(void) {
    std::cout << std::hex << transform(0x11223344);
    return 0;
}

int transform(int i) {
    int maske = 0xff, merke = i;
    switch (sizeof(int)) {
        case 2:
            *((byte *) &i + 1) = (byte) maske & merke;
            *((byte *) &i) = (byte) maske & (merke >> 8);
            break;
        case 4:
            *((byte *) &i + 3) = (byte) maske & merke;
            *((byte *) &i + 2) = (byte) maske & (merke >> 8);
            *((byte *) &i + 1) = (byte) maske & (merke >> 16);
            *((byte *) &i) = (byte) maske & (merke >> 24);
            break;
        default:
            std::cerr << "Fehler, integer Format nicht unterstützt!";
            exit(1);
    }
    return i;
}
```

Vorzuführen und zu erklären sind: dokumentierte C++-Quellprogramme. und durchgeführte Tests. Der Abgabetermin wird separat bekannt gegeben.

Aufgabenblatt 13 – Anwendung zyklische Warteschlange

Aufgabe 1

Programmieren Sie eine zyklische Warteschlange für C-Strings. Verwenden Sie ein `char *` Feld konstanter Länge (z.B. 13) für die Pointer die auf die C-Strings zeigen. Lesen Sie zum Einfügen eines Elements eine Zeichenkette von der Konsole ein (in ein Pufferfeld), ermitteln seine Länge, allokieren den entsprechenden Speicherplatz dynamisch und fügen den Eintrag in die Warteschlange ein. Entwickeln Sie Funktionen für:

1. Eintrag einfügen,
2. Eintrag entfernen,
3. Warteschlange auf der Konsole ausgeben,
4. Anzahl Einträge ausgeben,
5. Warteschlange neu initialisieren, d.h. alle Einträge entfernen und Warteschlange in den Anfangszustand versetzen.

Verwenden Sie ein Menü zum Testen. Für die Warteschlangenlogik können Sie ihre Ergebnisse aus Aufgabenblatt 12 Aufgabe 1 verwenden.

Erweitern Sie Ihr Programm, indem Sie einen zusätzlichen Menüpunkt „Warteschlange initialisieren“ hinzufügen, wobei Sie die Länge der erzeugten Warteschlange angeben können. Jetzt können Sie natürlich kein `char *` Feld konstanter Länge mehr deklarieren. Sie müssen das `char *` Feld dynamisch mit Hilfe von `new` erzeugen.

Modifizieren Sie Ihr Programm, indem Sie einen zusätzlichen Menüpunkt „Warteschlangen Länge ändern“ hinzufügen. Einträge in Ihrer Warteschlange sollen bei diesem Menüpunkt erhalten bleiben. Vergrößern der Warteschlange soll auf jeden Fall funktionieren. Verkleinern ist optional, d.h. muss nicht unbedingt implementiert werden.

Hinweis: Unter C-String verstehen wir ein `char`-Feld, das mit einem 0-Zeichen begrenzt ist. Denken Sie an die Funktionen zur Unterstützung der C-Stringbehandlung z.B. `strcpy()` oder `strlen()`.

Vorzuführen und zu erklären sind: dokumentiertes C++-Quellprogramm. Diese Aufgabe ist etwas schwieriger. Abgabemodus und Abgabetermin werden separat bekannt gegeben.